Objective-C & iOS Workshop

Patrick Jayet Oliver Gepp Sascha Thoeni

26th Sept. 2013

1 Stack

1.1 Create a New Project

Goal: TBD

TBD

1.2 Implement a Stack

Goal: TBD

TBD

2 UI

2.1 Linking Interface Items

Goal: in this exercise, we will see how to link items defined in the Interface Builder – classes, outlets and event methods – with the corresponding elements in the code.

1. Open the skeleton project RpnCalcPart2. The setup of the project comprises:
   - The model: a Stack and Computer model classes.
   - A storyboard file where the UI of the app is defined.
   - Various less important files.
2. Create a class CalcViewController, which should subclass UIViewController
   - As seen in the exercise 1.
   - The new class 'CalcViewController' should be a subclass of 'UIViewController'.
   - We have know an empty view controller where some default methods are already implemented
3. Link the controller class in the Interface Builder with the newly created controller class in the code
   - In the Project Navigator, click on the file 'Main.storyboard'. The Interface Builder is shown with the interface file.
   - On the left side of the Interface Builder, in the view hierarchy tree, click on the the file 'View Controller' (see screenshot 1)
   - On the right side select the Identity Inspector (see screenshot 1)
   - Under 'Custom Class', type the name of the newly created controller class 'CalcViewController'.
   - The link between the class in the code and in IB is done.

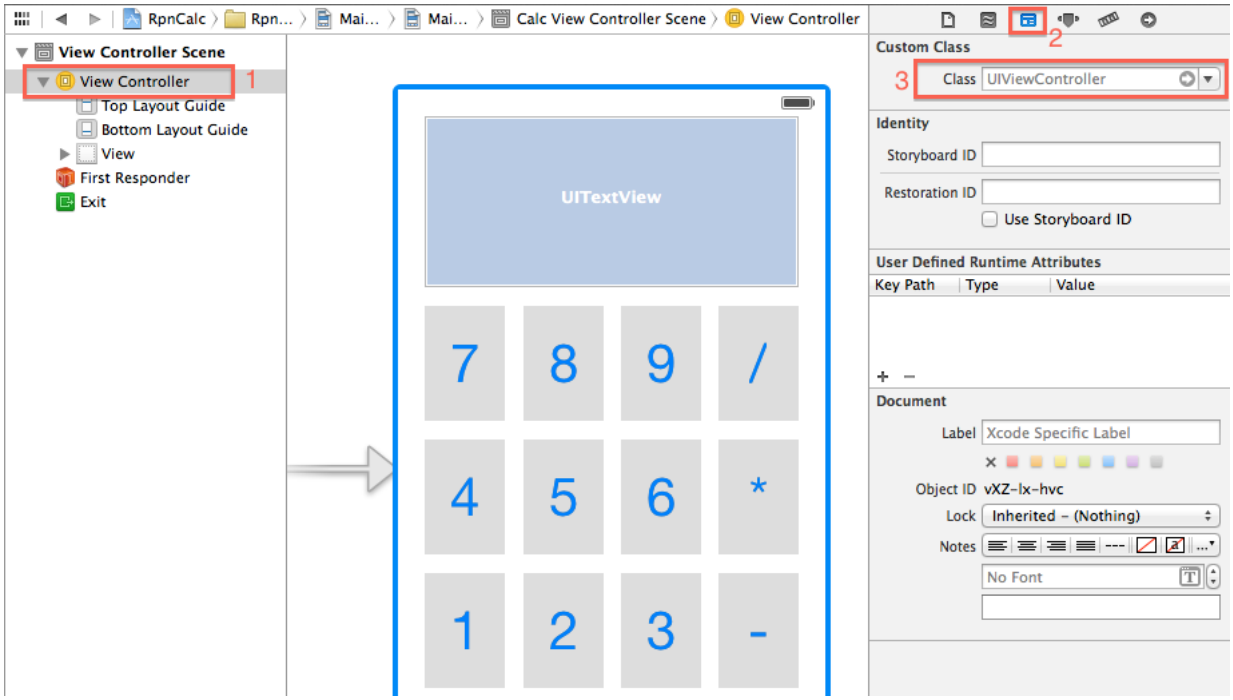

Figure 1: Screenshot of IB showing the Identity Inspector and Custom Class field.

4. Define an outlet for the result text view

   - In the 'CalcViewController' (i.e. in the header .h), add the following property between '@interface' and '@end':

     ```
     @property (strong) IBOutlet UITextView* resultView;
     ```

   - The prefix 'IBOutlet' does nothing in the code, except to indicate to IB that this property will be used as an UI item outlet.
   - Go to IB (select 'Main.storyboard').
   - Right click on 'Calc View Controller' in the left side, a dark gray panel appears, where we see the name of our result property 'resultView' (see screenshot 2)
   - Drag-click from the circle at the right of 'resultView' to the element 'UITextView' on the right (see screenshot 2).
   - Now the connection is done between the property in the code and the UI element in IB.

5. In 'CalcViewController', define UI callback methods.

   - In order to get events in the code when we press on the calculator keys, we do the connection with the desired events in IB.
   - First define in the controller three callback methods:

     ```
     - (IBAction)digitPressed:(id)sender {
         UIButton* button = sender;
         NSString* digit = button.currentTitle;
         NSLog(@"Digit pressed %@", digit);
         // further implementation
     }
     - (IBAction)operationPressed:(id)sender {
         UIButton* button = sender;
         NSString* operation = button.currentTitle;
         NSLog(@"Operation pressed %@", operation);
     ```

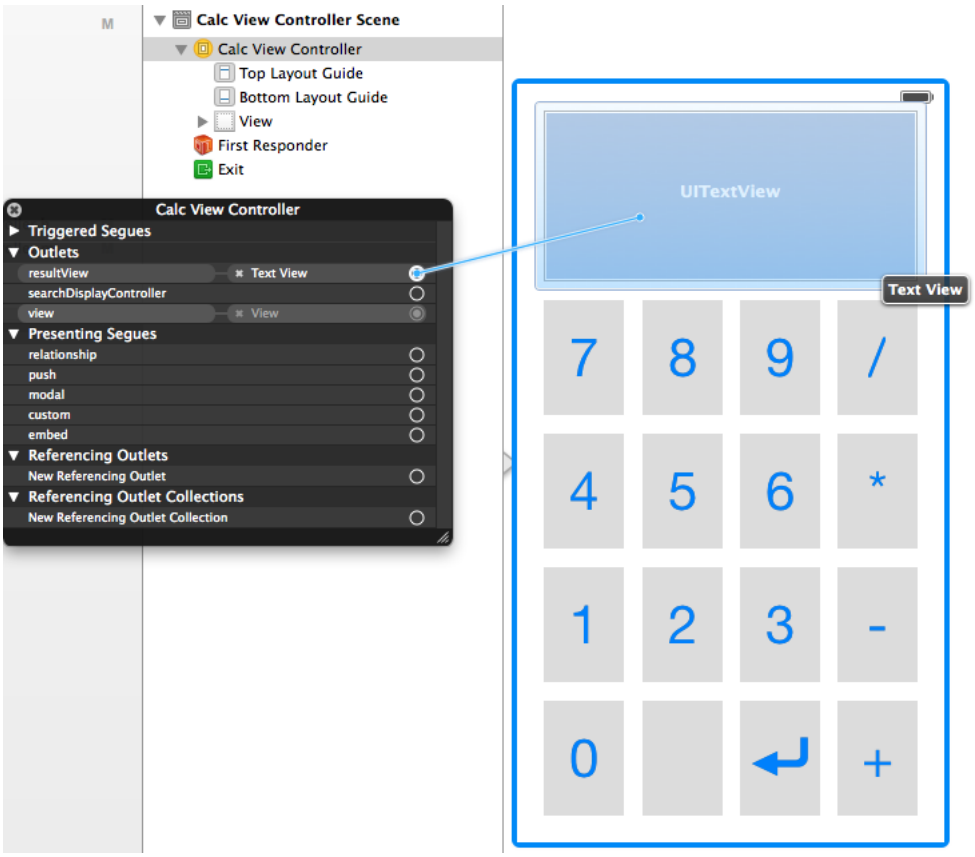

Figure 2: Connecting a view from IB with an outlet in the code.

```
    // further implementation
}
- (IBAction)enterPressed:(id)sender {
    NSLog(@"Enter pressed");
    // further implementation
}
```

- Similarly to the outlet definition, the 'IBAction' keyword does nothing in the code – it is a synonym to 'void' – except to indicate to IB that the corresponding methods are going to receive IB events. The 'sender' parameter is always the view where the event comes from – in this case an UIButton.
- Go to the IB, right click on a digit button, connect the event 'Touch Up Inside' with the controller 'Calc View Controller' on the left and select the method 'digitPressed:'. You have now connected one button. Do the same for the other digit buttons (to the same method).
- Now connect all four operator keys '+', '-', '*' and '/' to the method 'operationPressed:'. Here again the event 'Touch Up Inside'.
- Finally connect the enter key to the method 'enterPressed:'.

6. Now you can start the simulator (the play button on the top left of Xcode). You should see the calculator popping up. When you press on a key, you should see in the console the message of the corresponding callback method.

2.2 Linking the Business Logic

Goal: in this exercise, we will see how to call the business logic and refresh the UI from the callback methods in the controller.

1. In the controller (the .m file) define and initialize a private property

   - According to the following snippet – should be before the '@implementation' block.

     ```
     @interface CalcViewController ()
     @property (readwrite) Computer* computer;
     @end
     ```

   - Remember to import "Computer.h" at the top of the file, for the type 'Computer' to be visible.
   - In the method 'viewDidLoad', initialize the property. A property should always be accessed using the dot notation, e.g. 'self.foo'.

     ```
     self.computer = [[Computer alloc] init];
     ```

2. In the controller (the .m file), define a method to update the UI

   - Use the following snippet

     ```
     - (void)updateUI
     {
         // code here
     }
     ```

   - Complete the missing part. The method should:
     - Read the resulting text from the computer object.
     - Set the text content of the resultView – have a look at the text property of UITextView.

3. Call 'updateUI' from the end of 'viewDidLoad'.

4. Now complete the three callback methods 'digitPressed:', 'operationPressed:' and 'enterPressed:'.

   - Each method should do the following:
     - Call the corresponding method on the 'computer' property.

- Call the method to update the UI.

5. That's it. Everything should be in place now. Build and run the project. The calculator should work correctly.

   - Hint: this is a calculator which works in RPN mode – reverse polnish notation. To compute '1 + 2' you need the following steps:
     - Press '1'
     - Press 'Enter'
     - Press '2'
     - Press '+'

3 API from an External Framework & Gesture Recognizer

3.1 Say it!

Goal: TBD
TBD

3.2 Shake it!

Goal: TBD
TBD